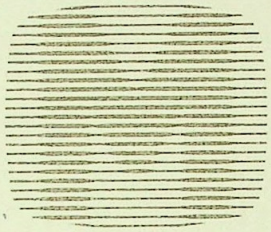

MICROWARE®

A/BASIC SOURCE GENERATOR

VERSION 2.0

REFERENCE MANUAL

Microware Systems Corporation
5835 Grand Avenue
Box 4865
Des Moines, Iowa 50304
515-279-8844

A/BASIC SOURCE GENERATOR V2.0 REFERENCE MANUAL

Copyright 1979 Microware Systems Corporation All Rights Reserved

Microware Systems Corporation distributes this manual for use by its licensees and customers. The information and programs contained herein is the property of Microware Systems Corporation and may not be reproduced or duplicated by any means without express written authorization.

SOFTWARE LICENSE AND LIMITED WARRANTY:

A/BASIC is copyrighted property of Microware Systems Corporation and is furnished to its customers for use on a single computer system owned by the customer. The program may not be copied in any form except for use on the customer's computer system. This license is not transferable and the customer may not make this software available to any third party without the express written consent of Microware Systems Corporation.

Purchase of this program and manual is considered implied consent of the provisions of the software license and warranty.

Microware Systems Corporation warrants this software to be free from defects for a period of 90 days after date of purchase. Any corrections may be made by means of printed or magnetic media at the option of Microware Systems Corporation.

Microware Systems Corporation reserves the right to make changes without notice (except within the warranty period) in the interest of product improvement at any time. Microware Systems Corporation cannot be responsible for any damages, including indirect or consequention, caused by reliance on this product's performance or accuracy of its documentation.

MICROWARE SYSTEMS CORPORATION
P.O. BOX 4865
DES MOINES, IOWA 50304
(515) 279-8844

SECOND EDITION

Printed in U.S.A.

INSTALLING THE SOURCE GENERATOR

Because versions of the A/BASIC Compiler and Source Generator are available for several different disk operating systems, this section will describe the general installation process, and not specific DOS commands, etc. A text file called SGINFO on the Source Generator disk has the specific commands and addresses for the particular operating system.

The object file SRCGEN supplied contains the complete compiler extensions which consist of the main Source Generator program and overlay patches to specific parts of the compiler.

The installation steps are:

1. Load the compiler program into the system memory. This is typically done with a GET system command.

2. Load the Source generator program into system memory. Parts of this file will overlay sections of the compiler.

3. Set the high memory limit for the compiler tables. Instructions for doing this are contained in the SGINFO file on the Source Generator disk.

4. The new combined program is then saved as a disk command file.

USING THE SOURCE GENERATOR

The new combined compiler/source generator program will function identically as the original A/BASIC Compiler unless the Source Generator is activated by the inclusion of an OPT H statement anywhere in the source A/BASIC program.

When activated, the source generator will generate an assembly language formatted listing and/or an assembly language output file, depending on other compile time options (OPT L, N, etc.).

The assembly language listing is identical in format to a typical 6800 assembler listing and is discussed in detail elsewhere in this manual.

If 'object' file parameters and/or name was given in the command line that invoked the compiler, an assembly language source output file will be created in place of the normal (machine language) object file.

This output file is not formatted except as required for assembler input syntax and is therefore similar to manually created assembler files. This output file is a standard text file which can be processed with the system's usual text editor and re-assembled source output file will produce machine language output identical to the compiler's output.

SOURCE GENERATOR OUTPUT

The assembly language output automatically produced by the Compiler/Source Generator corresponds to the machine language generated that will execute the A/BASIC source program.

The assembly language output has three sections:

1. System address definitions.
2. Program variable definitions.
3. Program instructions.

SYSTEM ADDRESS DEFINITIONS

These definitions use the assembler EQU directive to symbolically define addresses of:

a) Any BASLIB subroutines included in the program such as:

```
MULT EQU $024a
```

b) All the A/BASIC permanent run-time variables used by BASLIB or other routines. There are a total of eleven such variables, and all are included whether or not actually used in a particular program. Consult the A/BASIC Compiler manual for descriptions of their usage.

c) Addresses of operating system entry points used by the program such as:

```
OUTCHR EQU $E1D1
```

PROGRAM VARIABLE DEFINITIONS

These define the variables used by the specific compiled program. Some of these will correspond to A/BASIC variables and others may be compiler allocated variables.

To make the assembly language listing correspond to the source program as closely as possible, the Source Generator uses symbolic definitions that include the variable names used in the source A/BASIC program. There are several type of variables which are described below. In the definitions "n" corresponds to the original A/BASIC variable name.

NUMERIC VARIABLES are defined as VARn.
Example: A/BASIC variable X2 becomes VARX2.

STRING VARIABLES are defined as STRn.
Example: A/BASIC string B$ becomes STRB.

FOR/NEXT VARIABLES are allocated by the compiler for loop terminating values and for increment values if a STEP was given. These names are related to the loop counter variable name. The terminating variable is defined as LPn and the increment is defined as IVARn.
Example: The statement FOR Z=1 TO 25 STEP V/2 will cause the Source Generator to create variables LPZ and IVARZ.

FILE CONTROL BLOCKS. If the compiled program uses disk I/O statements, file control blocks are assigned the name FILEn where n is the channel number.

BUFFER DEFINITIONS. The names VARIO and VARST are defined to the starting addresses of the input/output and string buffers, respectively, if used by the program.

PROGRAM INSTRUCTION SECTION

This section is the actual assembly language program output. Each original A/BASIC source statement is included in the program as a comment followed by the assembly language instructions for the statement.

The program may use up to four types of statement labels:

LINE NUMBER LABELS have the form LNn where n is the line number used in the original A/BASIC program.
Example: The A/BASIC line number 250 will translate to LN250.

LOOP LABELS identify the top instruction of a FOR/NEXT loop and have the form LPn where n is a source generator assigned number.

INTERNAL LABELS are generally used in elements of IF/THEN statements and have the form LAn where n is a source generator assigned number.

STRING CONSTANT LABELS identify the starting address of string literals ("messages") in the program and have the form MGn where n is a source generator assigned number.

Special assembler directives such as NAM, ORG, and END are also automatically included in the output program.

MANUAL OPTIMIZATION OF A/BASIC PROGRAMS

The compiled assembly language output of the compiler/source generator can be edited with a text editor and re-assembled with a standard 6800 assembler. It is therefore possible to edit the program to increase its speed or decrease its size somewhat, though these goals often conflict.

Because A/BASIC is an optimizing compiler that has been designed to produce an optimum tradeoff between size and speed, there is only a limited amount of additional manual optimization that can be performed. Some possibilities in this area are given below:

1. Conversion of JMP/JSR instructions to BRA/BSR instructions when in range often reduce program size and increase speed. This optimization cannot be done by A/BASIC with only 2 passes so most programs can be optimized this way.

2. Subroutining of common instruction sequences can be done in some programs to decrease size at the expense of execution speed. Be careful is code segments contain labels!

3. Use of one-byte operations for variables which are known to always have a value of less than 255 can decrease size and increase speed. This must be done very carefully especially where the variable is used in mixed expressions or passed to BASLIB subroutines.

4. Replacement of divisions and multiplications by constant powers of two with shifting instructions. This will usually increase size but decrease execution time.

SAMPLE OUTPUT LISTING

```
                        NAM     DATST2.C
                OUT     EQU     $10B1
                ST1     EQU     $11A1
                CMD     EQU     $7783
                ODM     EQU     $7780
                ZWM     EQU     $7283
                FCBTMP  EQU     $10
                ERRADR  EQU     $12
                ERRTYP  EQU     $14
                PEKTMP  EQU     $17
                XSAVE   EQU     $20
                BUFPTR  EQU     $22
                ZONE    EQU     $26
                STRPTR  EQU     $27
                STRSAV  EQU     $29
                OVERFL  EQU     $2B
                XTEMP   EQU     $2D
                POSCNT  EQU     $2F
                VARX2   EQU     $0030
                STRA    EQU     $003A
                VARA    EQU     $008A
                VARN    EQU     $008C
                LFN     EQU     $008E
                IVARN   EQU     $0090
                VARIO   EQU     $0092
                VARST   EQU     $0113
                *
                *OPT S,H
                *
                *REM TEST OF A/BASIC SOURCE GENERATOR
                *
                *DIM X2(5),A$(80)
                *
                *STACK=$3000
                        ORG     $1000
1000 8E 30 00           LDS     #$3000
                *
                *ON ERROR GOTO 600
1003 CE 10 88           LDX     #$1088
1006 DF 12              STX     ERRADR
                *
                *100 A=0
1008 4F                 CLR A
1009 5F                 CLR B
100A D7 8B              STA B   VARA+1
100C 97 8A              STA A   VARA
                *
                *A$="CURRENT VALUE IS"
100E 20 11              BRA     *+$13
1010                    FCC     /CURRENT VALUE IS/
1020                    FCB     00
1021 CE 10 10           LDX     #MG1
1024 DF 27              STX     STRPTR
1026 CE 00 3A           LDX     #STRA
1029 C6 50              LDA B   #$50
102B BD 11 DC           JSR     ST1+$3B
                *
```

```
                        *FOR N=1 TO 20 STEP 2
102E 4F                         CLR A
102F C6 01                      LDA B   #$01
1031 D7 8D                      STA B   VARN+1
1033 97 8C                      STA A   VARN
1035 4F                         CLR A
1036 C6 14                      LDA B   #$14
1038 D7 8F                      STA B   LFN+1
103A 97 8E                      STA A   LFN
103C 4F                         CLR A
103D C6 02                      LDA B   #$02
103F D7 91                      STA B   IVARN+1
1041 97 90                      STA A   IVARN
                        *
                        *A=A+N
1043 D6 8B              LP1     LDA B   VARA+1
1045 96 8A                      LDA A   VARA
1047 DB 8D                      ADD B   VARN+1
1049 99 8C                      ADC A   VARN
104B D7 8B                      STA B   VARA+1
104D 97 8A                      STA A   VARA
                        *
                        *GOSUB 500
104F BD 10 6F                   JSR     LN500
                        *
                        *NEXT N
1052 D6 8D                      LDA B   VARN+1
1054 96 8C                      LDA A   VARN
1056 DB 91                      ADD B   IVARN+1
1058 99 90                      ADC A   IVARN
105A D7 8D                      STA B   VARN+1
105C 97 8C                      STA A   VARN
105E D0 8F                      SUB B   LFN+1
1060 92 8E                      SBC A   LFN
1062 2D 05                      BLT     LA1
1064 26 06                      BNE     LA2
1066 5D                         TST B
1067 26 03                      BNE     LA2
1069 7E 10 43           LA1     JMP     LP1
                        *
                        *STOP
106C 7E 72 83           LA2     JMP     ZWM
                        *
                        *500 PRINT A$,A
106F BD 10 C4           LN500   JSR     OUT+$13
1072 CE 00 3A                   LDX     #STRA
1075 C6 50                      LDA B   #$50
1077 BD 11 39                   JSR     OUT+$88
107A BD 10 CB                   JSR     OUT+$1A
107D D6 8B                      LDA B   VARA+1
107F 96 8A                      LDA A   VARA
1081 BD 11 4A                   JSR     OUT+$99
1084 BD 10 EB                   JSR     OUT+$3A
                        *
                        *RETURN
1087 39                         RTS
                        *
```

```
                        *600 PRINT "ERROR NUMBER ";ERR
1088 BD 10 C4   LN600   JSR     OUT+$13
108B 20 0E              BRA     *+$10
108D            MG2     FCC     /ERROR NUMBER /
109A                    FCB     00
109B CE 10 8D           LDX     #MG2
109E BD 11 37           JSR     OUT+$86
10A1 D6 14              LDA B   ERRTYP
10A3 4F                 CLR A
10A4 BD 11 4A           JSR     OUT+$99
10A7 BD 10 EB           JSR     OUT+$3A
                        *
                        *STOP
10AA 7E 72 83           JMP     ZWM
                        *
                        *END
10AD                    END

LOAD MAP:

OUT 10B1
ST1 11A1

ERRORS  000
VAR LEN $01E2
PGM LEN $0287
PGM END $1286

X2    0030 05 00
A$    003A 00 50
A     008A 00 00
N     008C 00 00
*     008E 00 00
N*    0090 00 00
IO    0092 00 00
ST    0113 00 00
```